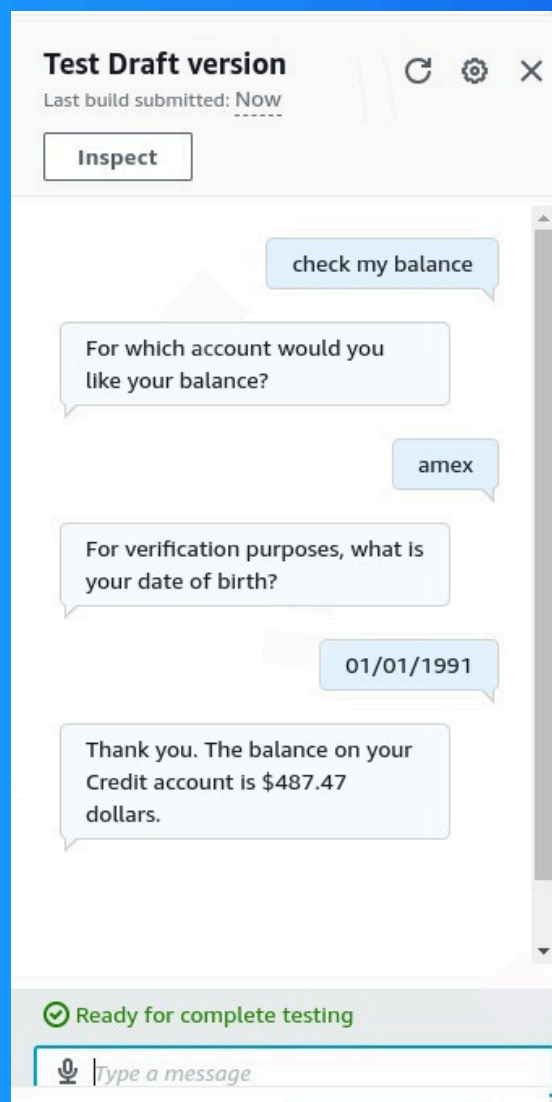




Connect a Chatbot with Lambda



Hassan Gachoka





Hassan Gachoka

[linkedin.com/gachokahassan](https://www.linkedin.com/in/gachokahassan)

NextWork.org

Introducing Today's Project!

What is Amazon Lex?

Amazon Lex is a service for building conversational interfaces using voice and text. It's useful for creating chatbots and virtual assistants, automating customer interactions, and integrating with AWS services for scalable, intelligent solutions.

How I used Amazon Lex in this project

In today's project, I used Amazon Lex to build a chatbot that interacts with users, processes their requests, and triggers AWS Lambda functions to return dynamic responses, such as fetching random bank balance figures.

One thing I didn't expect in this project was...

One thing I didn't expect in this project was how seamlessly Amazon Lex integrates with AWS Lambda to handle dynamic responses, making it easier to scale and customize the chatbot's behavior without complex infrastructure.

This project took me...

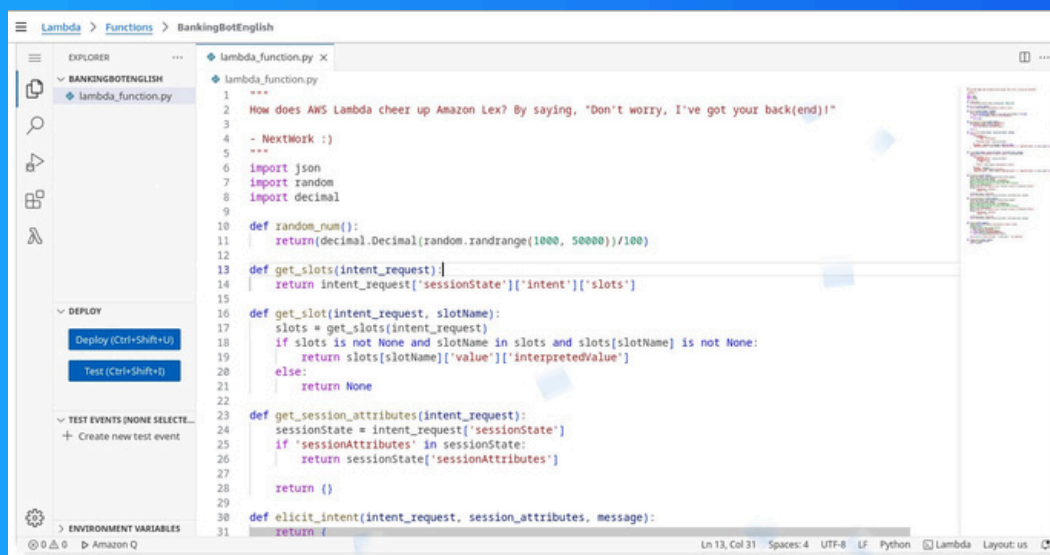
This project took me around 45 minutes, including setting up Amazon Lex, creating the Lambda function, and configuring the integration to return dynamic bank balance data based on user input.



AWS Lambda Functions

AWS Lambda is a serverless compute service that automatically runs your code in response to triggers like API requests, file uploads, or database events, scaling seamlessly without managing infrastructure.

In this project, I created a Lambda function to handle intents in Amazon Lex, like "CheckBalance" and "FollowupCheckBalance." It generates random account balances and sends personalized responses based on user input using session attributes.



```
1  """
2  How does AWS Lambda cheer up Amazon Lex? By saying, "Don't worry, I've got your back(end)!"
3
4  - NextWork :)
5  """
6  import json
7  import random
8  import decimal
9
10 def random_num():
11     return decimal.Decimal(random.randrange(1000, 50000))/100
12
13 def get_slots(intent_request):
14     return intent_request['sessionState']['intent']['slots']
15
16 def get_slot(intent_request, slotName):
17     slots = get_slots(intent_request)
18     if slots is not None and slotName in slots and slots[slotName] is not None:
19         return slots[slotName]['value']['interpretedValue']
20     else:
21         return None
22
23 def get_session_attributes(intent_request):
24     sessionState = intent_request['sessionState']
25     if 'sessionAttributes' in sessionState:
26         return sessionState['sessionAttributes']
27     return {}
28
29 def elicit_intent(intent_request, session_attributes, message):
30     return {
```



Chatbot Alias

An alias is a pointer to a specific version of an AWS Lambda function, enabling smoother deployment by referencing versions through a stable identifier. It helps manage function updates and rollbacks without directly interacting with version numbers.

TestBotAlias is an alias for an AWS Lambda function used with Amazon Lex. It links to a specific function version, ensuring consistent testing and development without affecting the primary version, allowing smooth updates and feature validation.

To connect Lambda with my BankerBot, I visited my bot's TestBotAlias and linked the Lambda function in the Languages section under "Source" and "Alias." This ensured the bot could interact seamlessly with the function for intent fulfillment.

Alias language support: English (US)

▼ **Lambda function - optional**
This Lambda function is invoked for initialization, validation, and fulfillment.

Source
BankingBotEnglish

Lambda function version or alias
\$LATEST

[Learn more about Lambda](#)

Cancel Save

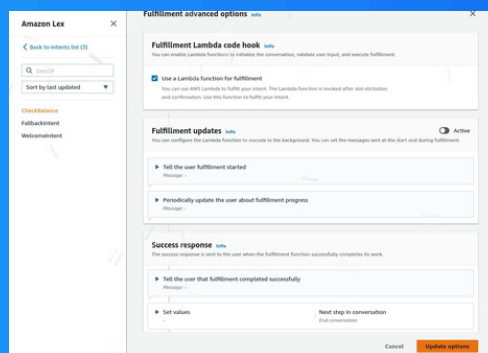


Code Hooks

A code hook is a mechanism in Amazon Lex that allows you to integrate Lambda functions to perform custom logic during the conversation. It triggers specific actions based on user inputs or intents, enabling dynamic and personalized responses.

Even though I connected my Lambda function with the chatbot's alias, I used code hooks because they allow customization of bot behavior, processing user inputs, and returning dynamic responses, such as generating a random bank balance.

I could find code hooks at the "CheckBalance" > "Fulfillment" > "Advanced options" > "Fulfillment Lambda code hook," where I linked the Lambda function to handle the fulfillment of the user's request for bank balance information.





Hassan Gachoka

linkedin.com/gachokahassan

NextWork.org

The final result!

I've set up my chatbot to trigger Lambda and return a random balance in dollar figure when the user requests a bank balance by specifying the account type, such as "CheckBalance" or "FollowupCheckBalance" intent.

